Package: ppn (via r-universe)

September 4, 2024

Encoding UTF-8

Type Package

Title Portable Piecepack Notation Parser

Version 0.1.0-2

Description Parse ``Portable Piecepack Notation'' files. This allows
you to visualize the moves for over one hundred board games.

License MIT + file LICENSE

URL https://github.com/piecepackr/ppn

BugReports https://github.com/piecepackr/ppn/issues

LazyLoad yes

Depends R (>= 3.4.0)

Imports affiner, bracer (>= 1.2), dplyr, ppdf (>= 0.1.0-4), rlang,
snakecase, stringr, tibble, utils, yaml

Suggests argparse, gifski, knitr, piecepackr (>= 1.10.1), ppcli,
shiny, testthat, vdiffr,

Remotes piecepackr/ppcli, piecepackr/ppdf, trevorld/affiner

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Config/testthat/edition 3

Repository https://piecepackr.r-universe.dev

RemoteUrl https://github.com/piecepackr/ppn

RemoteRef HEAD

RemoteSha cd97748b9e6e2d74275683f371d9ec83a91b44c8

Contents

`animate_game` *Animate a ppn game*

Description

Animate a ppn game

Usage

```
animate_game(
  game,
  file = "animation.gif",
  annotate = TRUE,
  ...,
  .f = piecepackr::grid.piece,
  cfg = NULL,
  envir = NULL,
  n_transitions = 0L,
  n_pauses = 1L,
  fps = n_transitions + n_pauses,
  width = NULL,
  height = NULL,
  ppi = NULL,
  new_device = TRUE,
  annotation_scale = NULL
)
```

Arguments

`game`	A list containing a parsed ppn game (as parsed by `read_ppn()`)
`file`	Filename to save animation unless `NULL` in which case it uses the current graphics device.
`annotate`	If `TRUE` or `"algebraic"` annotate the plot with “algrebraic” coordinates, if `FALSE` or `"none"` don't annotate, if `"cartesian"` annotate the plot with “cartesian” coordinates.
`...`	Arguments to `pmap_piece`
`.f`	Low level graphics function to use e.g. `grid.piece()`, `piece3d()`, `piece()`, or `piece_mesh()`.
`cfg`	A piecepackr configuration list
`envir`	Environment (or named list) of piecepackr configuration lists
`n_transitions`	Integer, if over zero (the default) how many transition frames to add between moves.
`n_pauses`	Integer, how many paused frames per completed move.

`fps`	Double, frames per second.
`width`	Width of animation (in inches). Inferred by default.
`height`	Height of animation (in inches). Inferred by default.
`ppi`	Resolution of animation in pixels per inch. By default set so image max 600 pixels wide or tall.
`new_device`	If `file` is `NULL` should we open up a new graphics device?
`annotation_scale`	Multiplicative factor that scales (stretches) any annotation coordinates. By default uses `attr(df, "scale_factor") %||% 1`.

Value

Nothing, as a side effect saves an animation of ppn game

See Also

`piecepackr::animate_piece()`

Examples

```
game_file <- system.file("ppn/tic-tac-toe.ppn", package = "ppn")
game <- read_ppn(game_file)[[1]]
if (require("gifski")) {
  animate_game(game, file = "tic-tac-toe.gif")
  unlink("tic-tac-toe.gif")
}
```

`cat_move` *View game in command-line terminal*

Description

`cat_move()` prints a plaintext diagram of a single move to the terminal. `cat_game()` prints a plaintext "animation" of every move to the terminal.

Usage

```
cat_move(game, move = NULL, ...)

cat_game(game, ..., fps = 1)
```

Arguments

`game`	A list containing a parsed ppn game (as parsed by `read_ppn()`)
`move`	Which move to cat game state (after the move, will use `game$dfs[[move]]`) unless `NULL` in which case will print the game state after the last move.
`...`	Passed to `ppcli::cat_piece()`.
`fps`	Frames per second.

plot_move *Plot game move*

Description

Plot game move

Usage

```
plot_move(
  game,
  file = NULL,
  move = NULL,
  annotate = TRUE,
  ...,
  .f = piecepackr::grid.piece,
  cfg = NULL,
  envir = NULL,
  width = NULL,
  height = NULL,
  ppi = 72,
  bg = "white",
  new_device = TRUE,
  annotation_scale = NULL
)
```

Arguments

`game`	A list containing a parsed ppn game (as parsed by `read_ppn()`)
`file`	Filename to save animation unless `NULL` in which case it uses the current graphics device.
`move`	Which move to plot game state (after the move, will use `game$dfs[[move]]`) unless `NULL` in which case will plot the game state after the last move.
`annotate`	If `TRUE` or `"algebraic"` annotate the plot with "algrebraic" coordinates, if `FALSE` or `"none"` don't annotate, if `"cartesian"` annotate the plot with "cartesian" coordinates.
`...`	Passed to `piecepackr::render_piece()`
`.f`	Low level graphics function to use e.g. `grid.piece()`, `piece3d()`, `piece()`, or `piece_mesh()`.
`cfg`	A piecepackr configuration list
`envir`	Environment (or named list) of piecepackr configuration lists
`width`	Width of animation (in inches). Inferred by default.
`height`	Height of animation (in inches). Inferred by default.

ppi	Resolution of animation in pixels per inch. By default set so image max 600 pixels wide or tall.
bg	Background color (`"transparent"`) for transparent
new_device	If `file` is `NULL` should we open up a new graphics device?
annotation_scale	Multiplicative factor that scales (stretches) any annotation coordinates. By default uses `attr(df, "scale_factor") %||% 1`.

Value

An invisible list of the dimensions of the image, as a side effect saves a graphic

---

read_ppn *Read PPN files*

Description

Read/write Portable Piecepack Notation (PPN) files

Usage

```
read_ppn(file, parse = TRUE)

write_ppn(games = list(), file = "")
```

Arguments

file	Filename, if "" will use `stdout()`
parse	Logical of whether to parse the moves in the ppn file
games	A list of parsed PPN games (as returned by `read_ppn()`)

Value

A list, for each game in the file a list containing info about the game

See Also

`plot_move()`, `animate_game()`, and `cat_move()` for visualizing parsed ppn games.

Examples

```
list.files(system.file("ppn", package = "ppn"))
file <- system.file("ppn/tic-tac-toe.ppn", package = "ppn")
games <- read_ppn(file)
tmp <- tempfile(fileext = ".ppn")
write_ppn(games, tmp)
unlink(tmp)
```

`view_game` *View/edit game*

Description

Launch PPN game viewer/editor

Usage

```
view_game(
  game,
  shiny = FALSE,
  ...,
  editor = getOption("editor"),
  reorient = "none",
  annotate = FALSE,
  fps = 1
)
```

Arguments

`game`	A list containing a parsed ppn game (as parsed by `read_ppn()`)
`shiny`	If `TRUE` launch a shiny PPN viewer in a browser instead of command-line viewer.
`...`	Passed to `plot_move()`.
`editor`	usually a character string naming (or giving the path to) the text editor you want to use.
`reorient`	Determines whether and how we should reorient (the angle) of pieces or symbols: 1. The default "none" (or `FALSE`) means don't reorient any pieces/symbols. 2. "all" (or `TRUE`) means setting the angle to zero for all pieces (reorienting them all "up"). 3. "symbols" means just re-orient suit/rank symbols but not the orientation of the piece itself. In particular, in contrast with "all" this preserves the location of the upper-left "corner" of piecepack tile faces.
`annotate`	If `TRUE` or `"algebraic"` annotate the plot with "algrebraic" coordinates, if `FALSE` or `"none"` don't annotate, if `"cartesian"` annotate the plot with "cartesian" coordinates.
`fps`	Frames per second. Passed to `cat_game()`.

Index